CodeSecure Architecture and Working

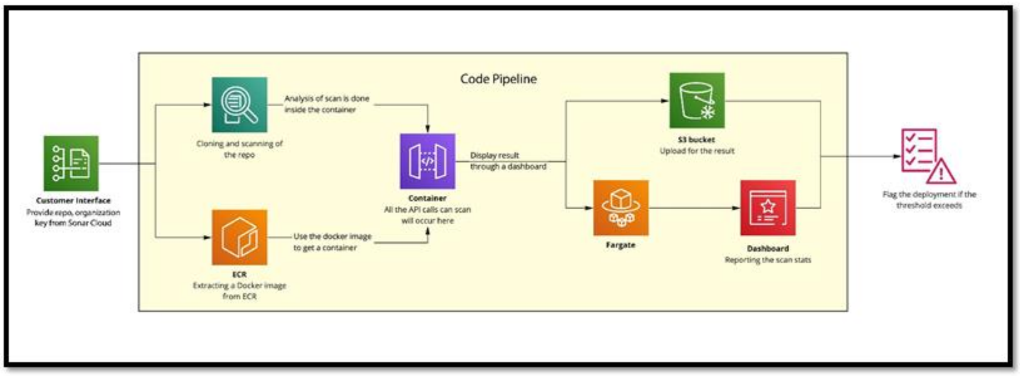

Code Pipeline
Customer Interface
Provide repo, organization key from Sonar Cloud
Cloning and scanning of the repo
Analysis of scan is done inside the container
ECR
Extracting a Docker image from ECR
Use the docker image to get a container
Container
All the API calls can scan will occur here
Display result through a dashboard
S3 bucket
Upload for the result
Fargate
Dashboard
Reporting the scan stats
Flag the deployment if the threshold exceeds

Figure-1

Architecture Explained:

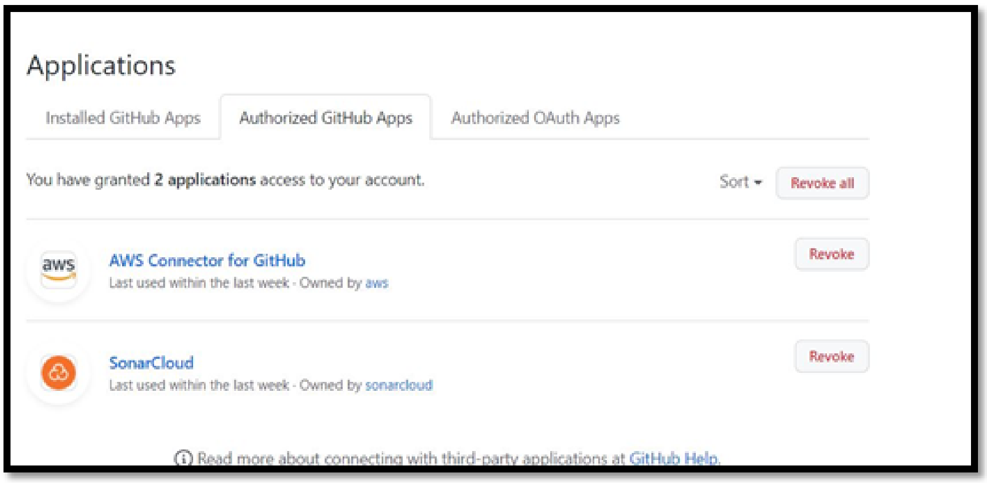
CodeSecure architecture is comprised of three stages, as shown in Figure-1. The first stage is the Customer interface stage. The pre-requisites are code repository such as Git/Bitbucket, and the second is the organisation key from Sonar Cloud. In this stage, CodeSecure-pipeline will take the commit from the code repository; once successful, it proceeds to the next stage.

The next stage is called the CodeSecure-scan or Code pipeline stage. Here, the code repository is cloned and scanned to generate the analysis report. In this stage, various technologies have been used to get the expected result. For example, the Docker image has been used to create a set of instructions using shell scripts. Amazon ECR has been used to extract the docker image. Containers are using this docker image to run the API calls for Sonarcloud and Fluid attacks. Amazon ECS has been used for managing these containers more logically. AWS load balancers are used to monitor the health of containers and services. Once the committed code or repository is analysed, it displays the result in two formats. One is in the report (PDF) format, which is stored in an Amazon S3 bucket. And another in the visual form, i.e., CodeSecure-dashboard. This dashboard has been created using Amazon Fargate service and Data Visualisation concepts. CodeSecure dashboard gives a better view and understanding of the issues, bugs and vulnerabilities the application may carry with it.

Working of CodeSecure:

CodeSecure is the proposed solution for SMB's who are dealing with IT (Information technology) applications. CodeSecure is using the AWS code pipeline service to implement its different stages, as explained in the product architecture (figure 1).

The pre-requisites required for organisations is to provide their code repository access such as Git / Bitbucket repository. For example, figure 2 shows the Git access for AWS and SonarCloud security tools.



Applications

Installed GitHub Apps | Authorized GitHub Apps | Authorized OAuth Apps

You have granted **2 applications** access to your account.

Sort ▾ | Revoke all

AWS Connector for GitHub
Last used within the last week · Owned by aws
Revoke

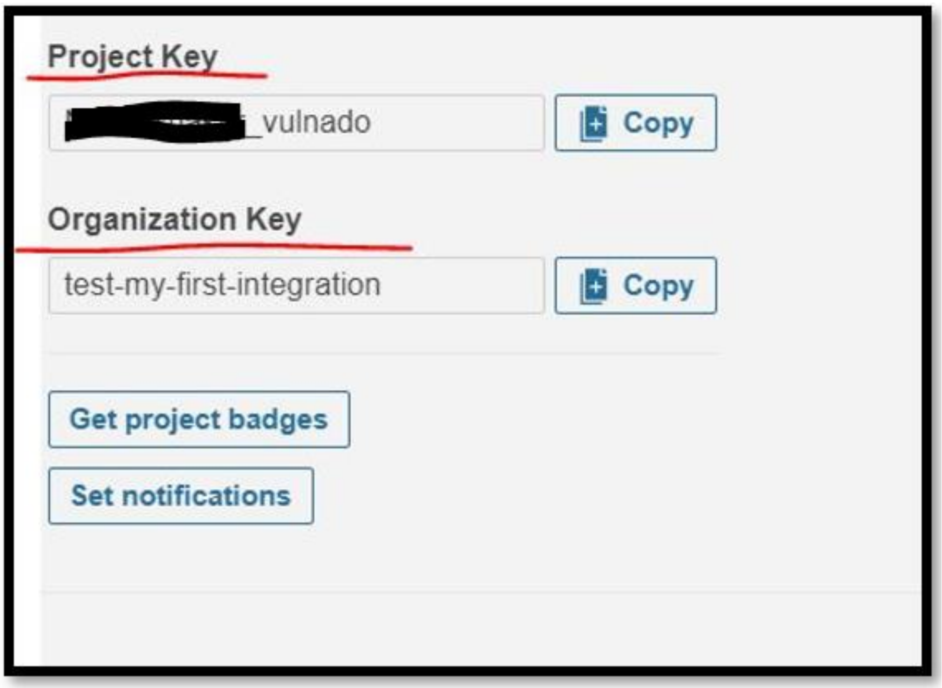
SonarCloud
Last used within the last week · Owned by sonarcloud
Revoke

ⓘ Read more about connecting with third-party applications at GitHub Help.

Figure-2

Apart from this, it is essential to note that the security tools that have been used in this application are considered keeping SMB's in mind. SonarCloud and Fluid attack provide free service to open projects. In case SMB's wants to keep the project environment hybrid or private only, these security tools are cost friendly as well.

The second requirement from the organisation is to provide the organisation key and project key for the Sonar Cloud, as shown in figure 3. These keys are used to get the result of the application using SonarCloud web API, which is used against Fluid attack reports.


Project Key

[redacted]_vulnado

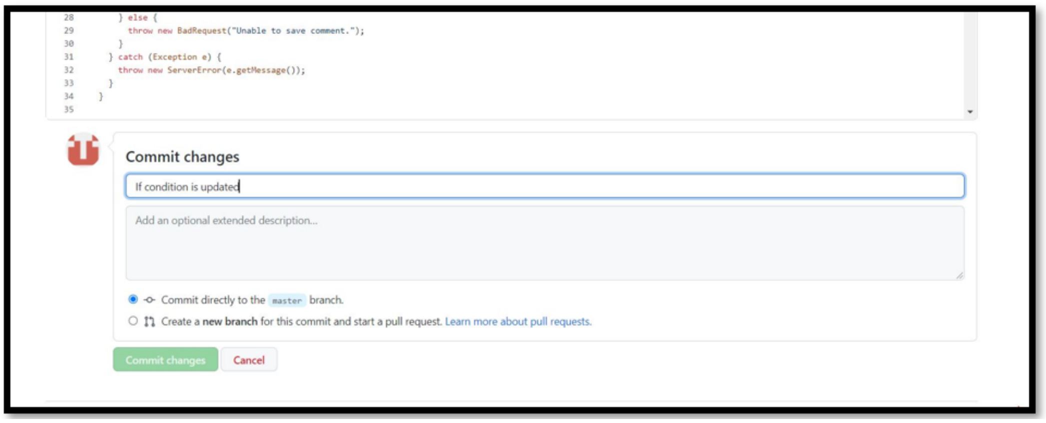
Organization Key

test-my-first-integration

<u>Figure-3</u>

The two security tools that have been used in the project are collectively giving their own analysis report on the applications or projects. In the CodeSecure pipeline, these analyses are combined and compared to give a better analysis report as well as a graphical representation of issues in the CodeSecure dashboard.

Once CodeSecure gets access to the pre-requisites, it starts with the first stage, which takes the commit from the Git repository. Once the commit is done or merged to master (figure 4), the CodeSecure pipeline's first stage, i.e. source stage, goes into the progress state.



```
      } else {
        throw new BadRequest("Unable to save comment.");
      }
    } catch (Exception e) {
      throw new ServerError(e.getMessage());
    }
  }

```

Commit changes

If condition is updated

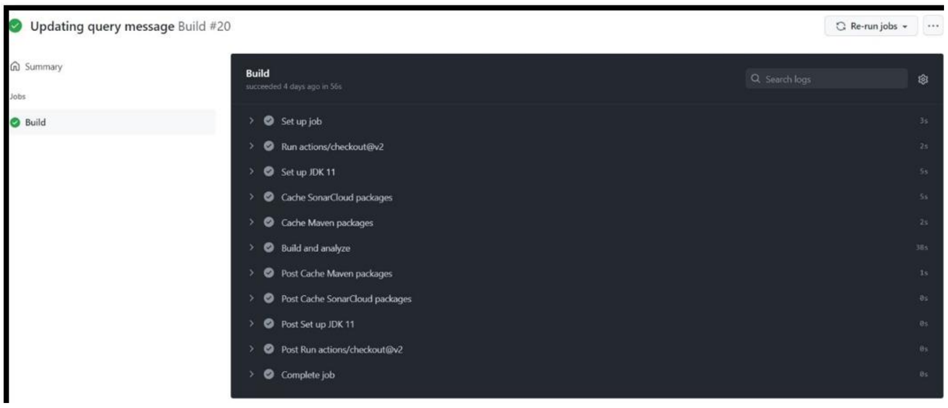
Add an optional extended description...

● Commit directly to the `master` branch.

○ Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

Figure - 4

In this stage, it builds and analyses the code (figure 5).



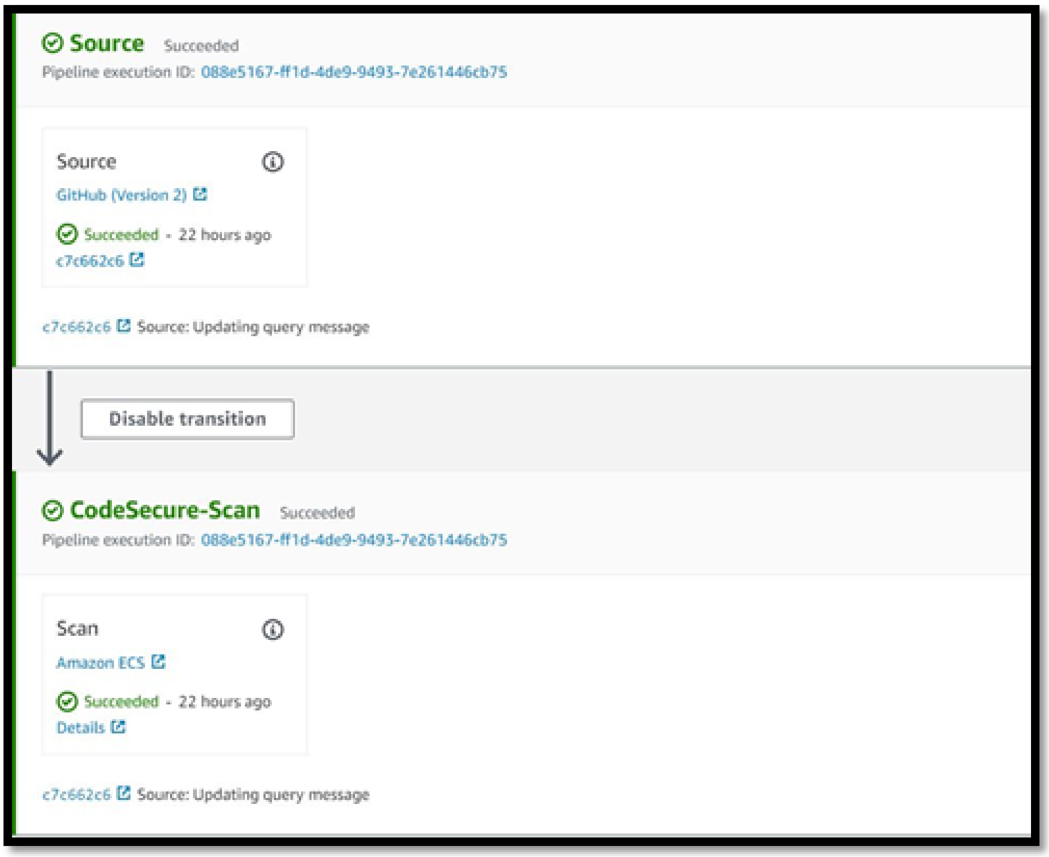
Updating query message Build #20

```
Build
succeeded 4 days ago in 56s

> Set up job                          3s
> Run actions/checkout@v2             2s
> Set up JDK 11                       5s
> Cache SonarCloud packages           5s
> Cache Maven packages                2s
> Build and analyze                   38s
> Post Cache Maven packages           1s
> Post Cache SonarCloud packages      0s
> Post Set up JDK 11                  0s
> Post Run actions/checkout@v2        0s
> Complete job                        0s
```

Figure – 5

As explained above, once it passes the first stage, it goes to the next stage i.e., CodeSecure-scan or Code pipeline (second stage), as shown in figure-6.



Source Succeeded
Pipeline execution ID: 088e5167-ff1d-4de9-9493-7e261446cb75

Source
GitHub (Version 2)
Succeeded - 22 hours ago
c7c662c6

c7c662c6 Source: Updating query message

CodeSecure-Scan Succeeded
Pipeline execution ID: 088e5167-ff1d-4de9-9493-7e261446cb75

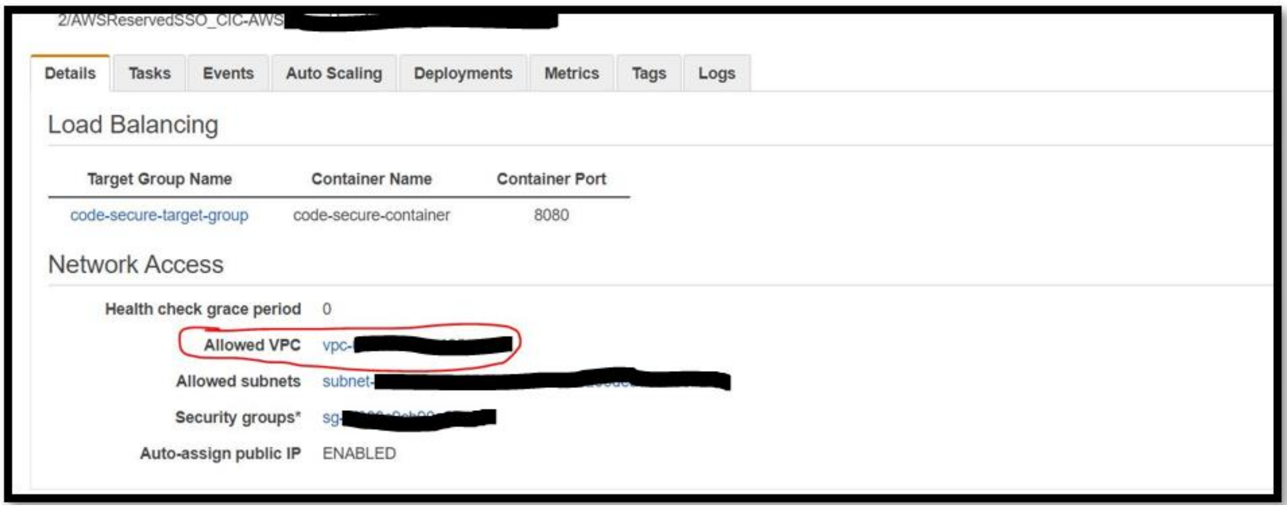
Scan
Amazon ECS
Succeeded - 22 hours ago
Details

c7c662c6 Source: Updating query message

Figure-6

In this stage, Amazon ECS service has been used, which provide clusters to manage the containers. Clustering can be explained as running one or more nodes (any physical machine or containers) in parallel to accomplish a task/s. In cloud computing, clusters are hosted using the Virtual private network (VPNs). VPNs can be seen in figure 7 and figure 8.

The reason the CodeSecure pipeline has used Amazon ECS for clustering is that with Amazon ECS, the scaling process is much easier, and it also includes other infrastructure benefits such as availability, reliability. With a cluster, it increases the latency by deploying the application instances in multiple nodes and on different availability zones [5].



2/AWSReservedSSO_CIC-AWS[redacted]

Details | Tasks | Events | Auto Scaling | Deployments | Metrics | Tags | Logs

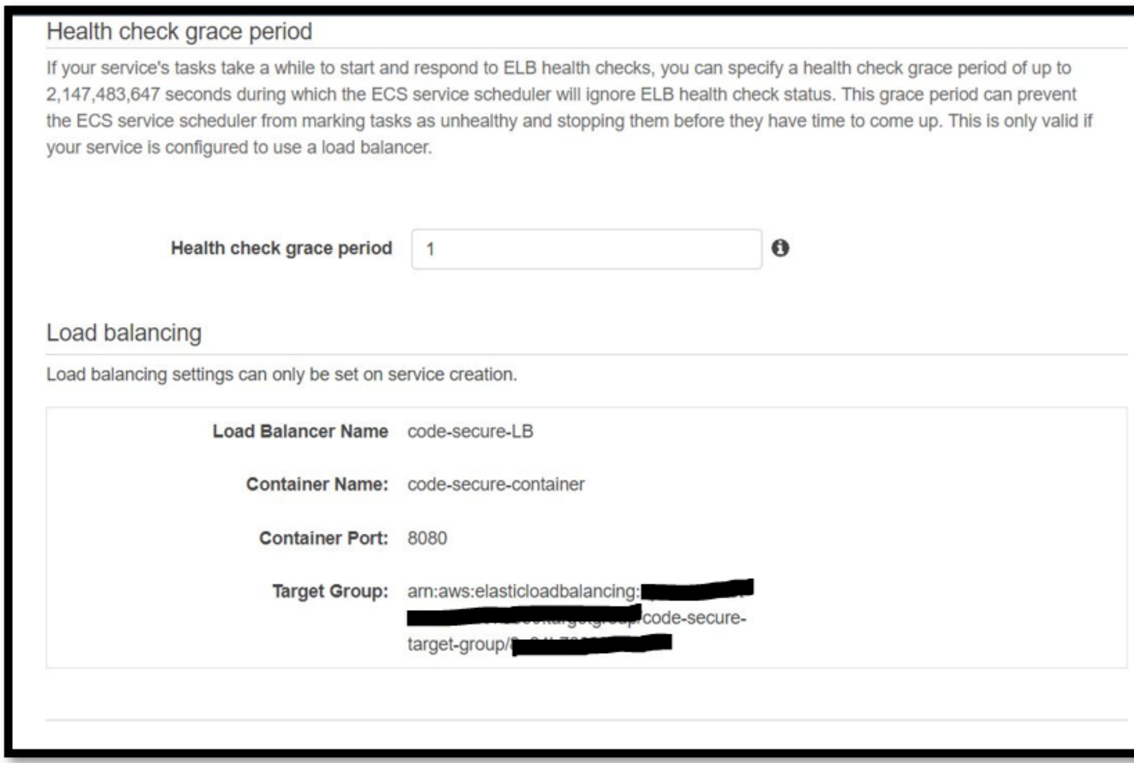
Load Balancing

Target Group Name	Container Name	Container Port
code-secure-target-group	code-secure-container	8080

Network Access

Health check grace period	0
Allowed VPC	vpc-[redacted]
Allowed subnets	subnet-[redacted]
Security groups*	sg-[redacted]
Auto-assign public IP	ENABLED

Figure-7



Health check grace period

If your service's tasks take a while to start and respond to ELB health checks, you can specify a health check grace period of up to 2,147,483,647 seconds during which the ECS service scheduler will ignore ELB health check status. This grace period can prevent the ECS service scheduler from marking tasks as unhealthy and stopping them before they have time to come up. This is only valid if your service is configured to use a load balancer.

Health check grace period: 1

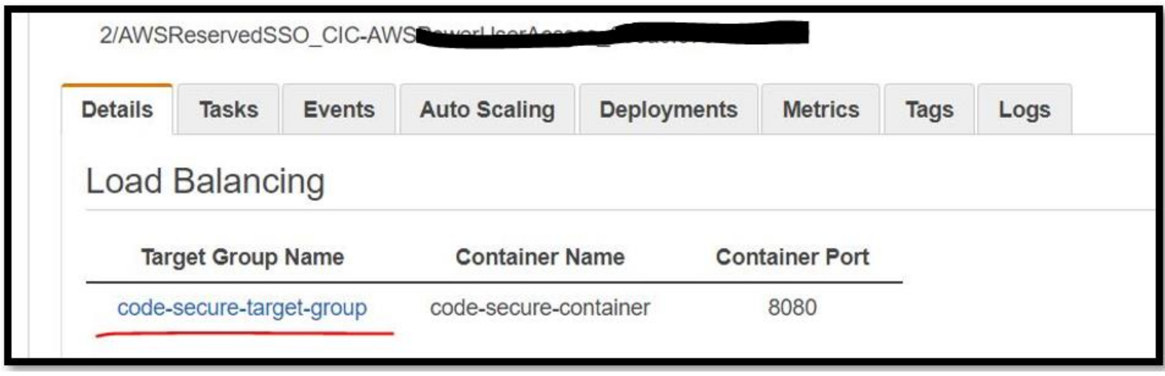
Load balancing

Load balancing settings can only be set on service creation.

Load Balancer Name	code-secure-LB
Container Name:	code-secure-container
Container Port:	8080
Target Group:	arn:aws:elasticloadbalancing:[redacted]code-secure-target-group/[redacted]

Figure – 8

Apart from VPNs, the service also uses Load balancers and Target groups to manage the health check of the services or nodes (which runs container as well). Load balancers can be seen in figure 8 with a health check of the service, and the target group can be seen in figure 11 and figure 12 with a similar health check of tasks or target groups.

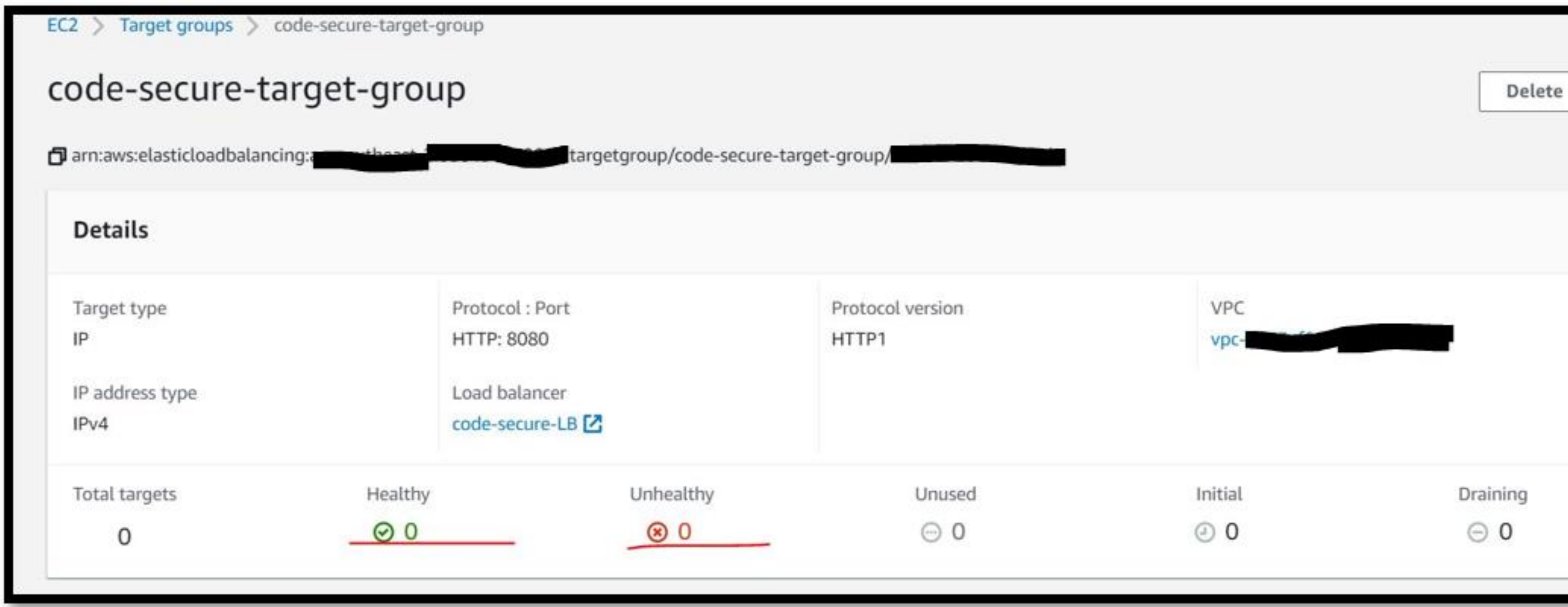

2/AWSReservedSSO_CIC-AWS[redacted]

Details | Tasks | Events | Auto Scaling | Deployments | Metrics | Tags | Logs

Load Balancing

Target Group Name	Container Name	Container Port
code-secure-target-group	code-secure-container	8080

Figure-9


EC2 > Target groups > code-secure-target-group

code-secure-target-group

Delete

arn:aws:elasticloadbalancing:[redacted]targetgroup/code-secure-target-group/[redacted]

Details

Target type IP	Protocol : Port HTTP: 8080	Protocol version HTTP1	VPC vpc-[redacted]
IP address type IPv4	Load balancer code-secure-LB		

Total targets	Healthy	Unhealthy	Unused	Initial	Draining
0	0	0	0	0	0

Figure – 10

Container, in simple terms, can be defined as a server inside another server and are primarily used to run docker image, which is a set of instructions in a script format. The reason container is used for the CodeSecure pipeline is because containers increase the security of applications by isolating them from the host. Complex application with microservice architecture also leverages the benefits of containerisation, as it eases the process of deployment. The following example illustrates the container and cluster relationship better [1]:

A web-based application was built using a microservices architecture. This means that the front and back-ends of the application will be standalone applications that communicate with each other over HTTPS. Such applications can be deployed using a virtual machine (VM) on the cloud. But this method has its flaws when application resources grow with time. Such a type of deployment over a VM can decrease the application performance as resources increase. The scaling process for such a methodology is also tricky and costly. It can even cause unavailability in the application if the VM or hardware on which it is hosted fails [1].

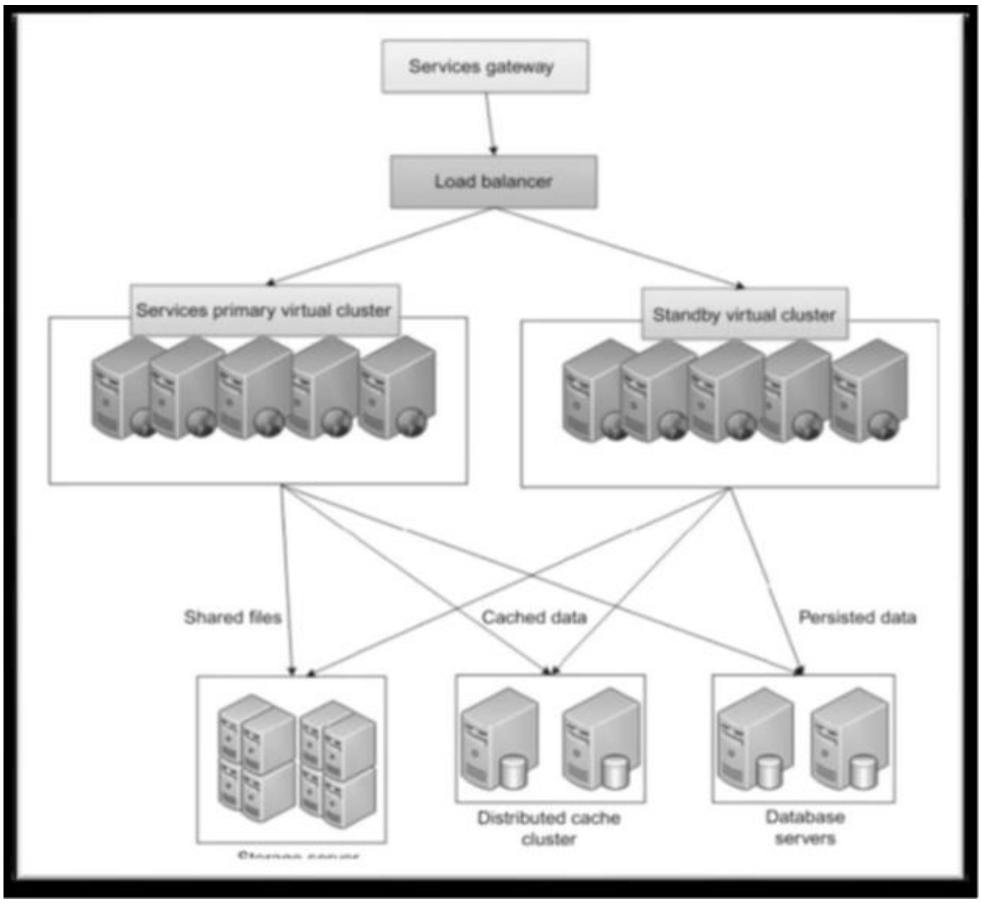
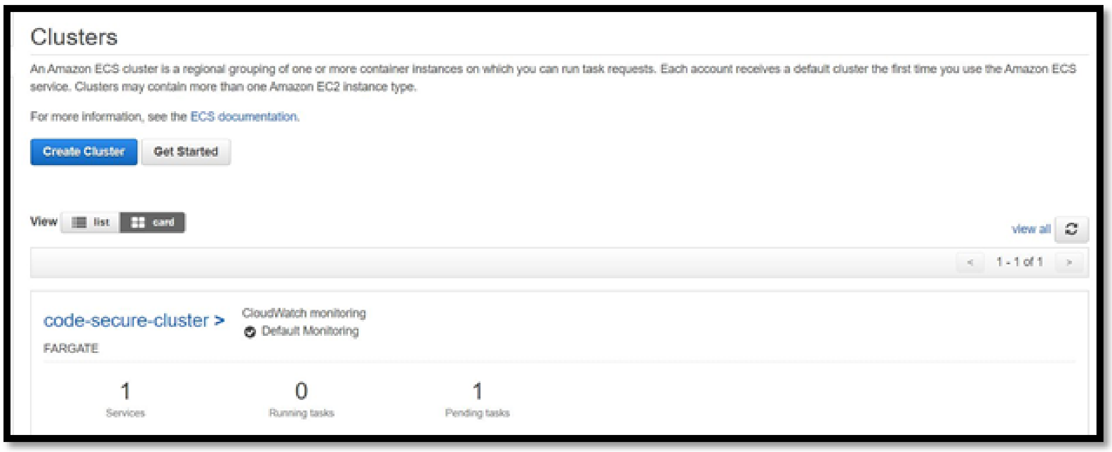


Figure – 11 [1]

This is where containerisation helps, and it even increases the benefits when these containers are deployed on clusters as it increases the availability, scalability and performance of the application. Figure 11 shows the structure of the universally available cluster. The reason Amazon ECS has been used for CodeSecure-pipeline is that it runs on multiple nodes, and these nodes are available in different geographical regions, which further increases the performance and availability of clusters [1]. Figure 14 shows the code-secure cluster.


Clusters

An Amazon ECS cluster is a regional grouping of one or more container instances on which you can run task requests. Each account receives a default cluster the first time you use the Amazon ECS service. Clusters may contain more than one Amazon EC2 instance type.

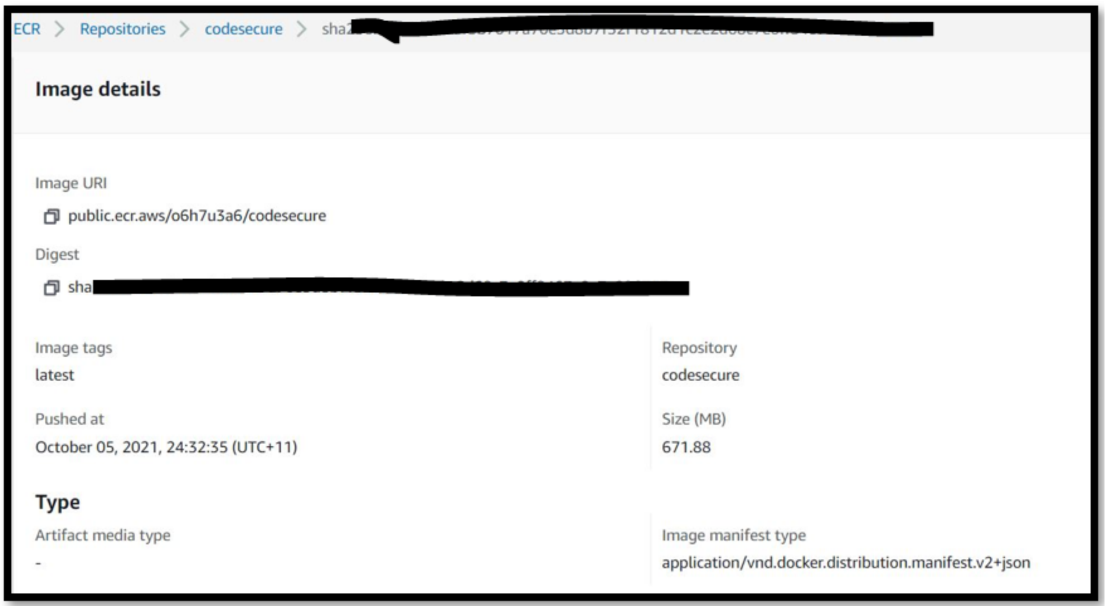
For more information, see the ECS documentation.

code-secure-cluster >
FARGATE
CloudWatch monitoring
Default Monitoring

1 Services
0 Running tasks
1 Pending tasks

Figure-12

As explained before, containers in the code-secure cluster have been used with docker images where some sets of instructions are implemented, and it is also used to call the API from SonarCloud and fluid attack security tools. Docker image has been uploaded to AWS cloud using Amazon ECR service, which can be shown in figure 15.



ECR > Repositories > codesecure > sha[redacted]

Image details

Image URI
public.ecr.aws/o6h7u3a6/codesecure

Digest
sha[redacted]

Image tags	Repository
latest	codesecure
Pushed at	Size (MB)
October 05, 2021, 24:32:35 (UTC+11)	671.88

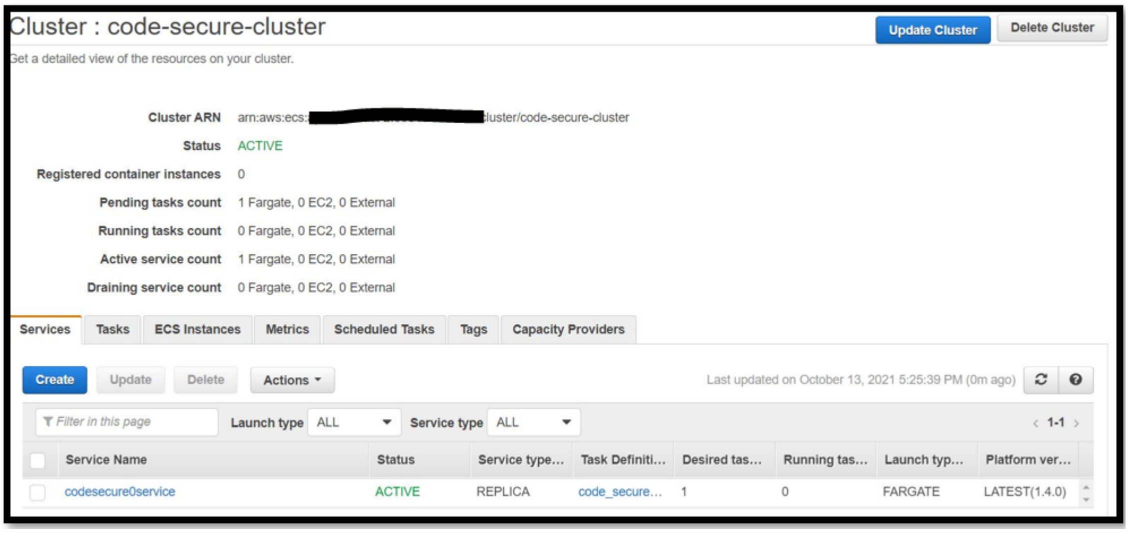
Type

Artifact media type	Image manifest type
-	application/vnd.docker.distribution.manifest.v2+json

Figure-13

For our CodeSecure pipeline, a cluster named code-secure-cluster has been created using the AWS Fargate service (figure 12). This cluster uses the services for the container management process. For example, we can define how many tasks can run inside a service.

Figure 14 shows the service named 'codesecure0service', and figure 15 shows the number of tasks the user wants to run simultaneously.



Cluster : code-secure-cluster

Get a detailed view of the resources on your cluster.

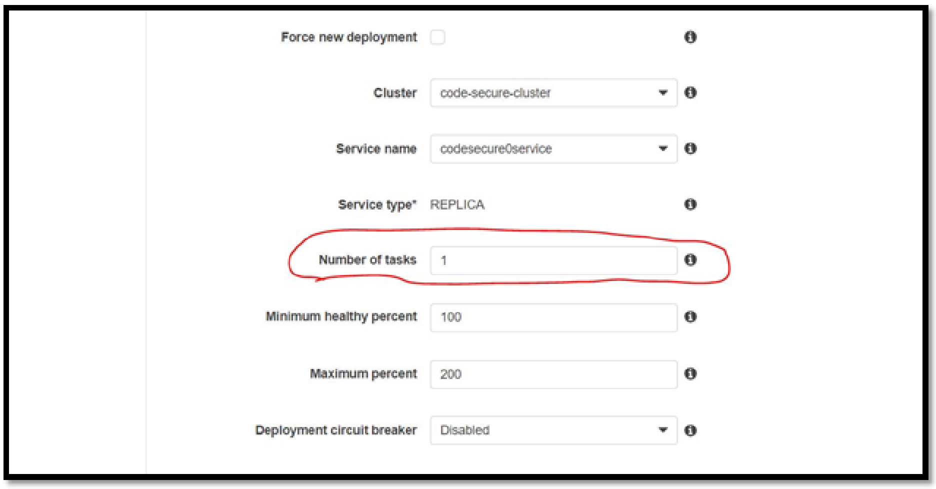
Cluster ARN	arn:aws:ecs:[redacted]luster/code-secure-cluster
Status	ACTIVE
Registered container instances	0
Pending tasks count	1 Fargate, 0 EC2, 0 External
Running tasks count	0 Fargate, 0 EC2, 0 External
Active service count	1 Fargate, 0 EC2, 0 External
Draining service count	0 Fargate, 0 EC2, 0 External

Last updated on October 13, 2021 5:25:39 PM (0m ago)

Launch type ALL   Service type ALL   1-1

Service Name	Status	Service type...	Task Definiti...	Desired tas...	Running tas...	Launch typ...	Platform ver...
codesecure0service	ACTIVE	REPLICA	code_secure...	1	0	FARGATE	LATEST(1.4.0)

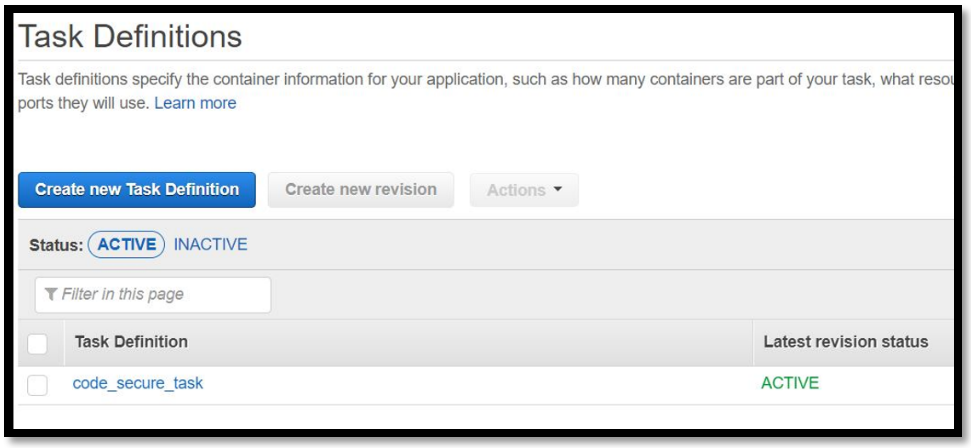
Figure-14



Force new deployment	☐
Cluster	code-secure-cluster
Service name	codesecure0service
Service type*	REPLICA
Number of tasks	1
Minimum healthy percent	100
Maximum percent	200
Deployment circuit breaker	Disabled

Figure-15

To run these tasks, a task definition has been created, as shown in figure 16. Task definition can have a single task or multiple tasks defined in it as per the user/organisation requirement. For example, figure 17 shows two active tasks with distinct characteristics or properties.

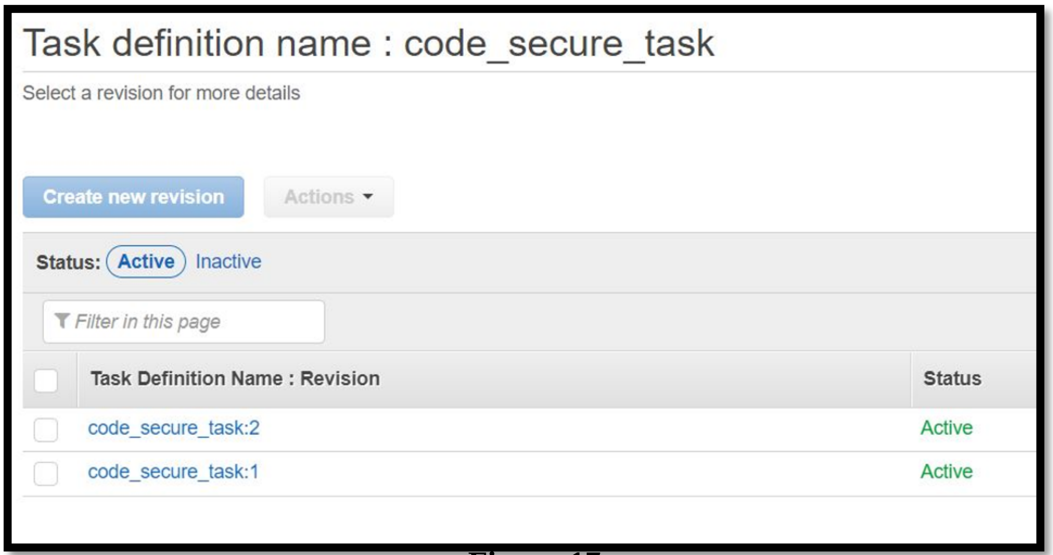


Task Definitions

Task definitions specify the container information for your application, such as how many containers are part of your task, what resou ports they will use. Learn more

Task Definition	Latest revision status
code_secure_task	ACTIVE

Figure-16



Task definition name : code_secure_task

Select a revision for more details

Create new revision | Actions

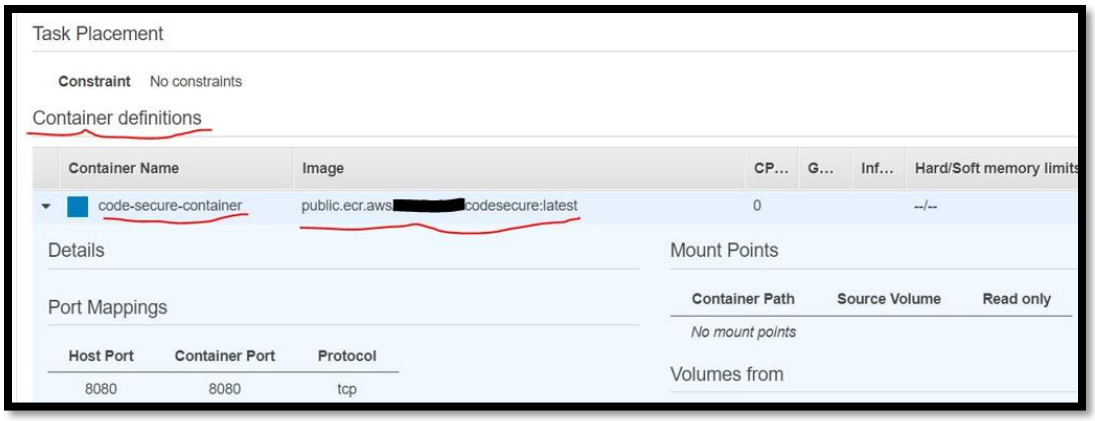
Status: Active Inactive

Filter in this page

Task Definition Name : Revision	Status
code_secure_task:2	Active
code_secure_task:1	Active

Figure-17

In these tasks, containers are created by providing the docker image URI (figure 18). Since these tasks are under a task definition, it is best practice to create one task definition per application (or container).



Task Placement

Constraint: No constraints

Container definitions

Container Name	Image	CP...	G...	Inf...	Hard/Soft memory limits
code-secure-container	public.ecr.aws[redacted]codesecure:latest	0			--/--

Details

Port Mappings

Host Port	Container Port	Protocol
8080	8080	tcp

Mount Points

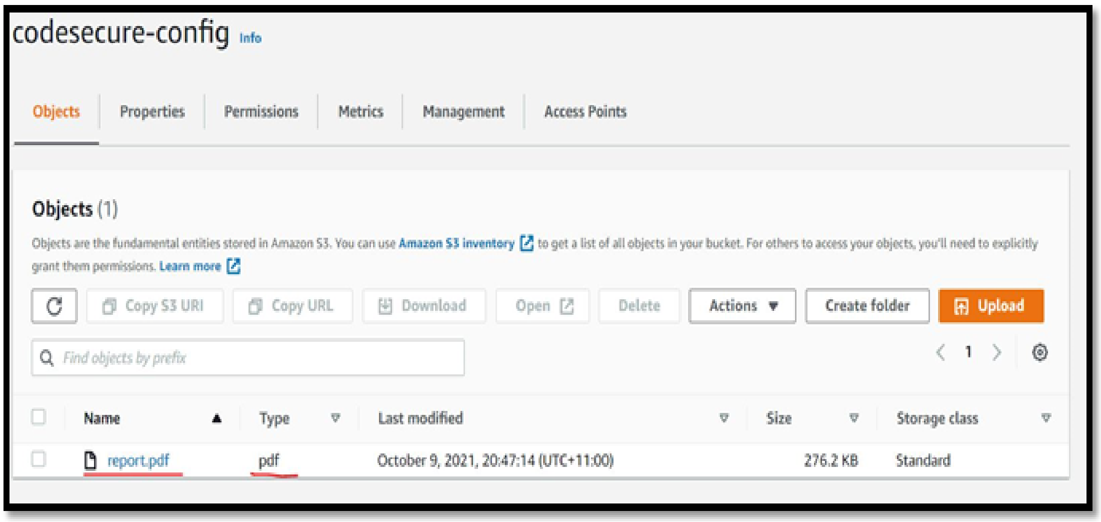
Container Path	Source Volume	Read only
No mount points		

Volumes from

Figure-18

Now, once the container's task is completed, i.e., code analysis is done, then it will create a Report.pdf file and store the file in the Amazon S3 bucket - "codesecure-config" (figure 19). The reason the Amazon S3 bucket is used for CodeSecure as it provides good encryption and strengthens the security of data while it is at rest or transit. Amazon S3 also offers secure key management, i.e., it provides an option for users to manage the keys when providing administrative options over data.

Amazon S3 service is also available in every region, hence increases the availability and performance [1].


codesecure-config Info

Objects | Properties | Permissions | Metrics | Management | Access Points

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
report.pdf	pdf	October 9, 2021, 20:47:14 (UTC+11:00)	276.2 KB	Standard

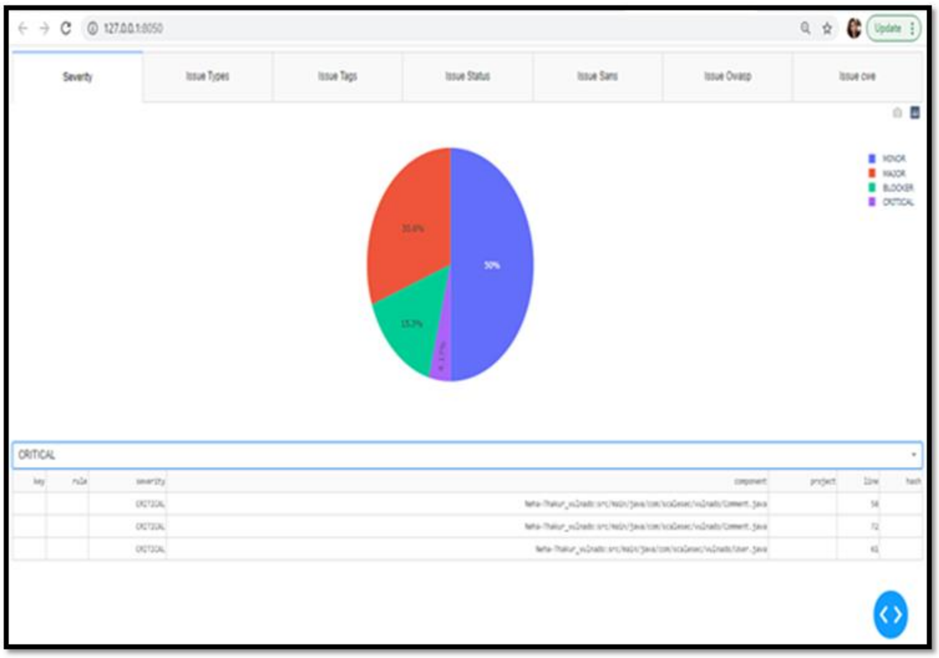
Figure-19

An example of the report.pdf file can be found in the GitHub repository named ‘report’.

CodeSecure dashboard gives the graphical representation of the analysis of the applications' issues/vulnerabilities/bugs. It also shows the status of the issues, i.e., if the issue has been resolved, closed or open, or top 10 OWASP issues. The dashboard also has a table like report.pdf, but the difference is that users can filter the issue types based on characteristics. For example, Characteristic – Severity and type are Major, Minor, Critical or Blocker.

The dashboard has used the data science concepts such as data visualisation and data wrangling to present the graphical representation of the final analysis report.

The CodeSecure – dashboard can be seen in figure 20.

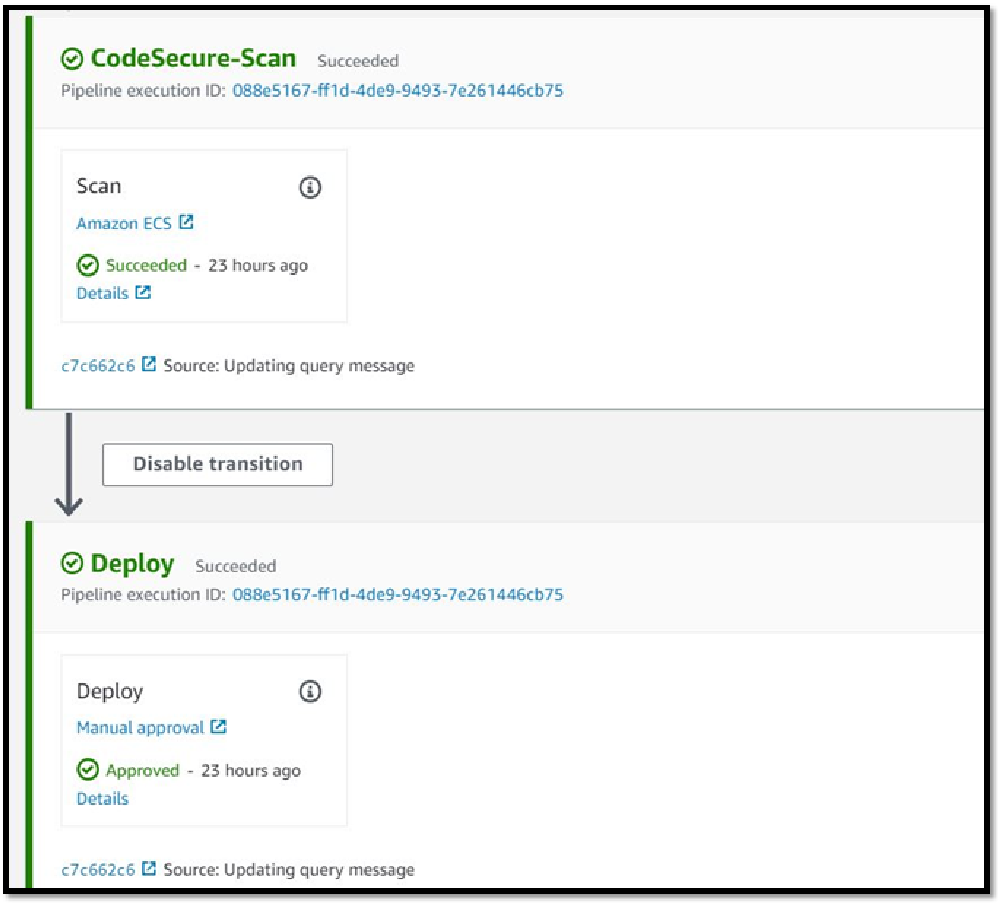


CRITICAL

key	rule	severity	component	project	line	hash
		CRITICAL	Neha-Thakur_vulnado:src/main/java/com/scalesec/vulnado/Comment.java		56	
		CRITICAL	Neha-Thakur_vulnado:src/main/java/com/scalesec/vulnado/Comment.java		72	
		CRITICAL	Neha-Thakur_vulnado:src/main/java/com/scalesec/vulnado/User.java		42	

<u>Figure – 20</u>

Now, once the CodeSecure-scan is completed (figure 21), the user can review the changes (figure 22) and based on the analysis report of the application, the user can manually approve or reject the deployment process (figure 26) in stage 3 of the code pipeline or CodeSecure – pipeline.



CodeSecure-Scan Succeeded
Pipeline execution ID: 088e5167-ff1d-4de9-9493-7e261446cb75

Scan
Amazon ECS
Succeeded - 23 hours ago
Details

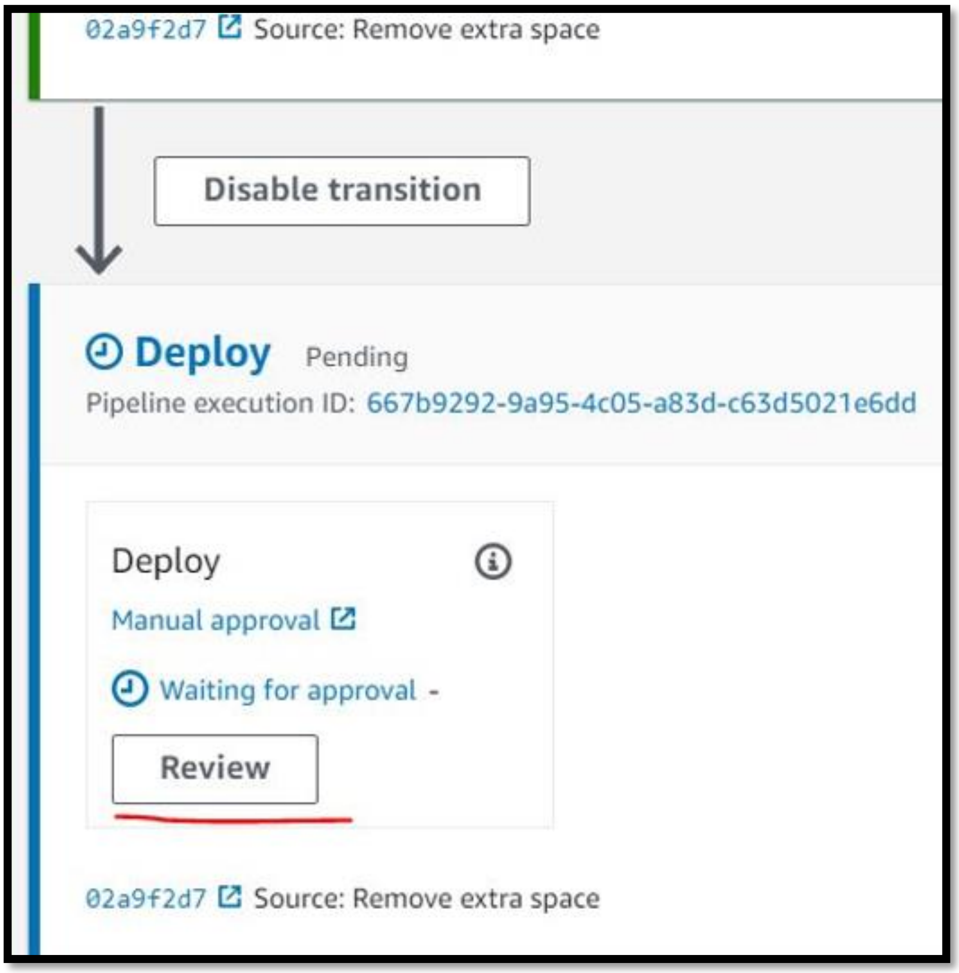
c7c662c6 Source: Updating query message

Deploy Succeeded
Pipeline execution ID: 088e5167-ff1d-4de9-9493-7e261446cb75

Deploy
Manual approval
Approved - 23 hours ago
Details

c7c662c6 Source: Updating query message

Figure-21


02a9f2d7 Source: Remove extra space

Deploy Pending

Pipeline execution ID: 667b9292-9a95-4c05-a83d-c63d5021e6dd

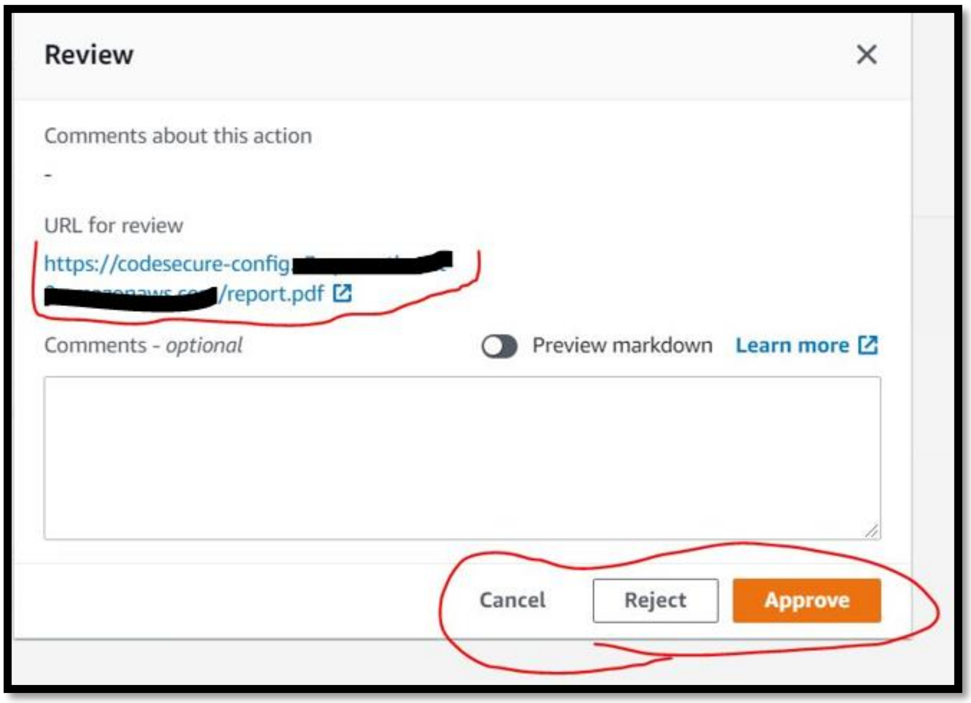
Deploy

Manual approval

Waiting for approval -

02a9f2d7 Source: Remove extra space

Figure-22



Review

Comments about this action

-

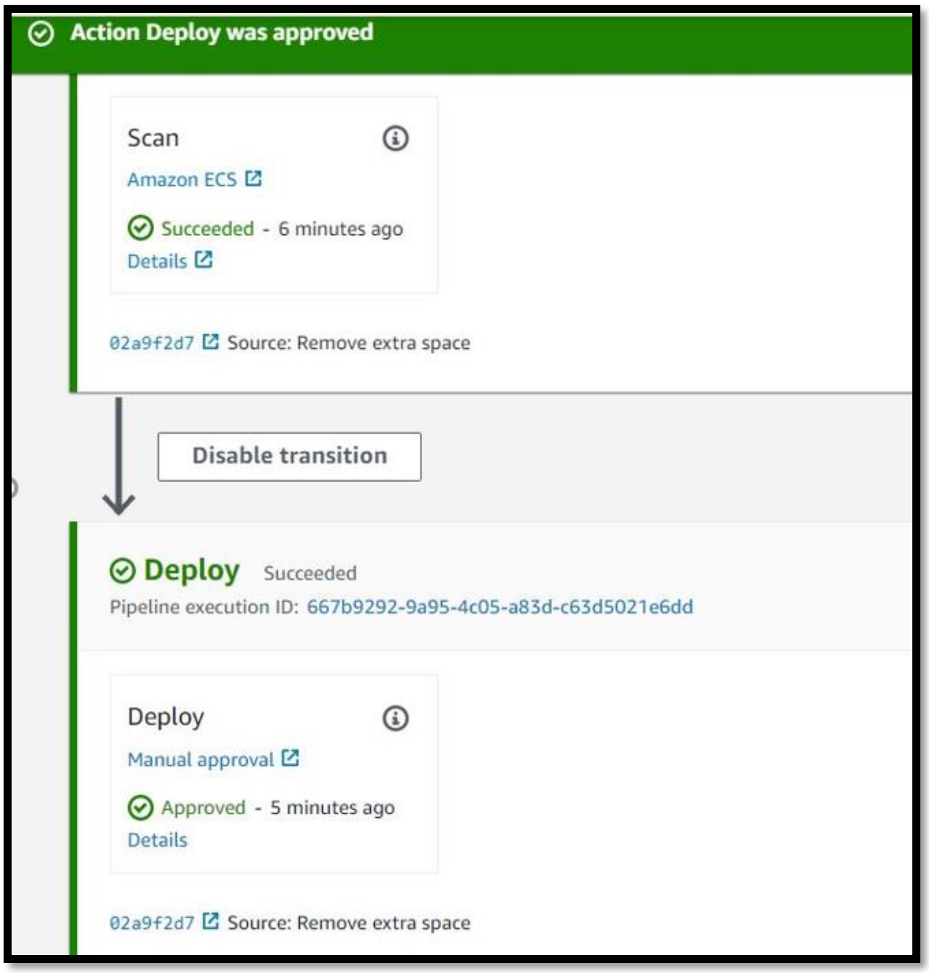
URL for review

https://codesecure-config[redacted]
[redacted]/report.pdf

Comments - *optional*

Figure-23

If the user rejects the deployment process, the CodeSecure-pipeline shows the last stage as failed (figure 24), otherwise approved (figure 25).



Action Deploy was approved

Scan
Amazon ECS
Succeeded - 6 minutes ago
Details

02a9f2d7 Source: Remove extra space

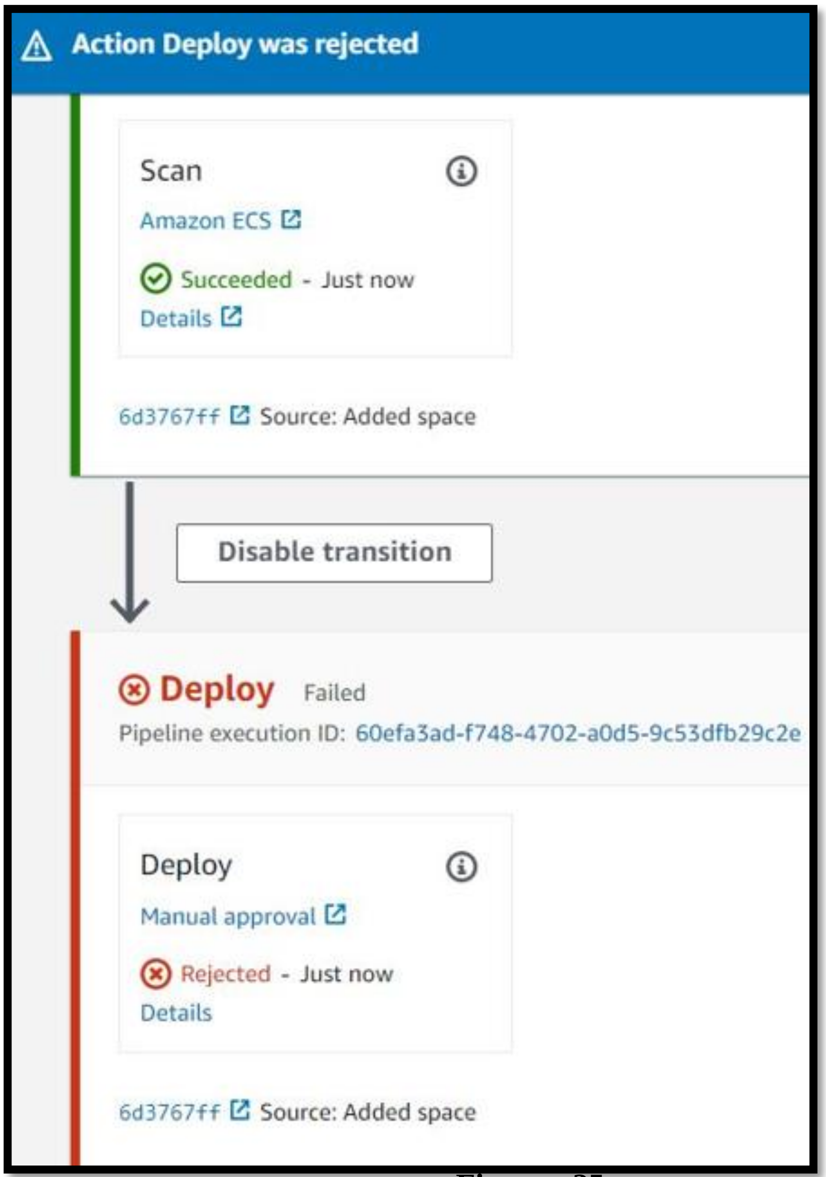
Disable transition

Deploy Succeeded
Pipeline execution ID: 667b9292-9a95-4c05-a83d-c63d5021e6dd

Deploy
Manual approval
Approved - 5 minutes ago
Details

02a9f2d7 Source: Remove extra space

Figure-24



Action Deploy was rejected

Scan
Amazon ECS
Succeeded - Just now
Details

6d3767ff Source: Added space

Disable transition

Deploy Failed
Pipeline execution ID: 60efa3ad-f748-4702-a0d5-9c53dfb29c2e

Deploy
Manual approval
Rejected - Just now
Details

6d3767ff Source: Added space

Figure - 25

References:

[1] Gajus Jonas Polikaitis, Neha Thakur, Tharindu Prabhashwara Wanninayake, "The Risks and Opportunities of Cloud-Based Infrastructure", unpublished, May 2021